Lab 4 Report – Gem 5 and McPAT

Kunjian Song

1 Automation Script using Gem5 and McPAT

The following study scripts were used to conduct design-space exploration experiments using Gem5 and McPAT:

- Makefile to compile the benchmark programs (*matmul.c* and *crc32.c*) using 2 different levels of optimization, -O0 and -O3.
- Run scripts to automate the performance experiments using Gem5 with 3 different memory models and 4 different operating frequencies.
- Run scripts to automate the power experiments for the above configurations

The memory models selected for the experiments are listed below:

- SimpleMemory
- DDR3_2133_8x8
- DDR4_2400_4x16

The clock frequencies selected for the experiments are listed below:

- 200 MHz
- 800MHz
- 1400MHz
- 2000MHz

The number of combinations for each benchmark program is 2*3*4 = 24 simulations. Since there are 2 benchmark programs and 2 types of simulations for each benchmark (performance and power), the total number of experiments that need to be run is 24*2*2 = 96 simulations. Hence, the Run scripts are used in this case to automate tasks. The following figures shows code snippets for Makefile.

```
FILE = crc32.c

bitcnts: ${FILE} Makefile
    arm-linux-gnueabi-gcc -static ${FILE} -O0 -marm -march=armv7-a -o crc32_O0_arm32
    arm-linux-gnueabi-gcc -static ${FILE} -O3 -marm -march=armv7-a -o crc32_O3_arm32
clean:
    rm -rf crc32_* output*
```

Figure 1: Makefile that compiles the benchmark program using -O0 and -O3 optimization level. This is a modified version of the provided Makefile script in the Ubuntu image.

As for the run scripts, bash script and functions are used to automate the simulation tasks. The main logic are shown in Figure 2.

```
1  #!/bin/sh
2
3  clear
4  echo ""
5  echo " Running gem5 simulations for ex1-COMP1"
6  echo ""
7
8  ########################## Run gem5 simulations #####################
9
10 cd /home/user/gem5/
11
12 run () {
13
14     OUTPUT=$1
15     CLK_FREQ=$2
16     MEM_TYPE=$3
17     BINARY=$4
18     POWER=$5
19
20     echo "Running with $1, $2, $3, $4, power: $5"
21
22      ./build/ARM/gem5.opt --stats-file=${OUTPUT} \
23         ./configs/example/se.py --sys-voltage="${POWER}V" \
24         --cpu-type=TimingSimpleCPU  \
25         --cpu-clock="${CLK_FREQ}MHz" \
26         --mem-type=${MEM_TYPE} \
27         -c /home/user/COMP61232/matmul/${BINARY}
28
29     echo "###"
30     echo "# Stage finished..."
31     echo "###"
32 }
33
34 # ==================== Un-optimized ========================
35 run "ex1_O0_SimpleMemory_200MHz.txt" "200" "SimpleMemory" "matmul_O0_arm32" "0.9"
36 run "ex1_O0_SimpleMemory_800MHz.txt" "800" "SimpleMemory" "matmul_O0_arm32" "1.1"
37 run "ex1_O0_SimpleMemory_1400MHz.txt" "1400" "SimpleMemory" "matmul_O0_arm32" "1.4"
38 run "ex1_O0_SimpleMemory_2000MHz.txt" "2000" "SimpleMemory" "matmul_O0_arm32" "1.9"
39
40 run "ex1_O0_DDR4_200MHz.txt" "200" "DDR4_2400_4x16" "matmul_O0_arm32" "0.9"
41 run "ex1_O0_DDR4_800MHz.txt" "800" "DDR4_2400_4x16" "matmul_O0_arm32" "1.1"
42 run "ex1_O0_DDR4_1400MHz.txt" "1400" "DDR4_2400_4x16" "matmul_O0_arm32" "1.4"
43 run "ex1_O0_DDR4_2000MHz.txt" "2000" "DDR4_2400_4x16" "matmul_O0_arm32" "1.9"
```

Figure 2: Automation script used for Gem5 simulations. Note that this is just a snippet to show the main logic. There are more lines to test DDR3 memory configuration and O3 optimization levels.

Function run() was designed to take various configuration arguments for Gem5. Line 14 – 18 are the input parameters for run() function. This function is called in Lin33 and onwards to run the simulation with various frequencies, memory systems and compiler optimization levels. Note that in Line 23 a voltage parameter is given to Gem5 when simulation using various frequencies.

Similar automation strategy are used for McPAT simulations. Figure 3 shows the code snippet for convert() and run_mcpat() functions used to automation McPAT simulations.

```
#!/bin/sh

clear

convert () {

    INPUT=$1
    OUTPUT=$2
    CLK_FREQ=$3

    echo "@@DEBUG: Converting with $1, $2, $3"
    python /home/user/COMP61232/gem5_to_mcpat.py /home/user/gem5/m5out/${INPUT} \
           /home/user/mcpat/ProcessorDescriptionFiles/${OUTPUT} \
           ${CLK_FREQ}

}

run_mcpat () {

    IN_XML=$1
    OUT_TXT=$2

    ./mcpat -infile ./ProcessorDescriptionFiles/${IN_XML} -print_level 0 > ./ProcessorDescriptionFiles/${OUT_TXT}

    echo " "
    echo "============================== Done with IN: $1. OUT: $2 ============================"
    echo "======= "

}
```

Figure 3: Functions used to convert gem5 output files and run McPAT simulations

The convert() and run_mcpat() functions are called as shown in Figure 4 and Figure 5 below:

```
echo " Converting Gem5 Stats files to McPAT XML files... "
#============================ O0 ============================
convert "ex1_O0_SimpleMemory_200MHz.txt"  "ex1_O0_SimpleMemory_200MHz.xml" 200
convert "ex1_O0_SimpleMemory_800MHz.txt" "ex1_O0_SimpleMemory_800MHz.xml" 800
convert "ex1_O0_SimpleMemory_1400MHz.txt" "ex1_O0_SimpleMemory_1400MHz.xml" 1400
convert "ex1_O0_SimpleMemory_2000MHz.txt" "ex1_O0_SimpleMemory_2000MHz.xml" 2000

echo "Done converting simple memory"

convert "ex1_O0_DDR4_200MHz.txt"  "ex1_O0_DDR4_200MHz.xml" 200
convert "ex1_O0_DDR4_800MHz.txt" "ex1_O0_DDR4_800MHz.xml" 800
convert "ex1_O0_DDR4_1400MHz.txt" "ex1_O0_DDR4_1400MHz.xml" 1400
convert "ex1_O0_DDR4_2000MHz.txt" "ex1_O0_DDR4_2000MHz.xml" 2000

echo "Done converting DDR4"
```

Figure 4: Automating the convert tasks

```
echo " Running McPAT Power Simulations... "
cd /home/user/mcpat
#============================ O0-MCPAT ============================
run_mcpat "ex1_O0_SimpleMemory_200MHz.xml" "ex1_mcpat_O0_SimpleMemory_200MHz.txt"
run_mcpat "ex1_O0_SimpleMemory_800MHz.xml" "ex1_mcpat_O0_SimpleMemory_800MHz.txt"
run_mcpat "ex1_O0_SimpleMemory_1400MHz.xml" "ex1_mcpat_O0_SimpleMemory_1400MHz.txt"
run_mcpat "ex1_O0_SimpleMemory_2000MHz.xml" "ex1_mcpat_O0_SimpleMemory_2000MHz.txt"

run_mcpat "ex1_O0_DDR4_200MHz.xml" "ex1_mcpat_O0_DDR4_200MHz.txt"
run_mcpat "ex1_O0_DDR4_800MHz.xml" "ex1_mcpat_O0_DDR4_800MHz.txt"
run_mcpat "ex1_O0_DDR4_1400MHz.xml" "ex1_mcpat_O0_DDR4_1400MHz.txt"
run_mcpat "ex1_O0_DDR4_2000MHz.xml" "ex1_mcpat_O0_DDR4_2000MHz.txt"

run_mcpat "ex1_O0_DDR3_200MHz.xml" "ex1_mcpat_O0_DDR3_200MHz.txt"
run_mcpat "ex1_O0_DDR3_800MHz.xml" "ex1_mcpat_O0_DDR3_800MHz.txt"
run_mcpat "ex1_O0_DDR3_1400MHz.xml" "ex1_mcpat_O0_DDR3_1400MHz.txt"
run_mcpat "ex1_O0_DDR3_2000MHz.xml" "ex1_mcpat_O0_DDR3_2000MHz.txt"
```

Figure 5: Automating the McPAT simulations using the converted files

2 Data Extraction and Processing Methods

The key data extraction targets are listed below:

- Time: “sim_seconds” form Gem5 output files

- Power: “Runtime Dynamic” from McPAT output files

Bash grep command was used for data extraction. The following figures show the command used for extracting “sim_seconds” and “Runtime Dynamic” data:

```
user@ubuntu:~/gem5/m5out$ grep "sim_seconds" ex1_O0_SimpleMemory_* | grep '[0-9]\.[0-9]\+' -oh
0.569242
0.555431
0.557666
0.557664
```

Figure 6: Bash command used to extract sim_seocnds

```
user@ubuntu:~/mcpat/ProcessorDescriptionFiles$ grep "Processor:" -a8 ex1_mcpat_O0_SimpleMemory_* | grep "Runtime Dynamic" | grep '[0-9]\.[0-9]\+' -oh
0.00378404
0.00527874
0.000817221
0.00230396
```

Figure 7: Bash command used to extract "Runtime Dynamic"

The main strategy used here is bash command grep with regular expression to extract the data. Note that the data extracted are in the order {1400MHz, 2000MHz, 200MHz, 800MHz} due to the file name sorting logic using grep. The option “-a8” was used to extract “Runtime Dynamic” power because we only want to extract the target line in “Processor: ” sections, rather than the lines below “Total Cores: 1 cores:” or “Total First Level Directory:”.

The data are copied into Google spreadsheet to generate the table and generate graph. An example is shown below:

Freq	SimpleMem_t	SimpleMem_O3	DDR3_t	DDR3_2133_8x8_O3	DDR4_t	DDR4_2400_4x16_O3	O3
1400	0.111264	8.987633017	0.148015	6.75607202	0.151175	6.614850339	
2000	0.10857	9.210647509	0.145262	6.884112844	0.148302	6.742997397	
200	0.109673	9.118014461	0.151184	6.614456556	0.152514	6.556775116	
800	0.10966	9.119095386	0.147347	6.786700781	0.151093	6.618440298	
Freq	SimpleMem_p	SimpleMem_O0	DDR3_p	DDR3_2133_8x8_O0	DDR4_p	DDR4_2400_4x16_O0	O0
1400	0.00378404	0.002154034498	0.00372317	0.002627061306	0.00371524	0.002705902173	
2000	0.00527874	0.002931975837	0.0052114	0.003657219812	0.00520705	0.003717380687	
200	0.000817221	0.0004557363662	0.000745544	0.0005349852269	0.000741172	0.0005413149702	
800	0.00230396	0.001284835549	0.0022369	0.001576063818	0.00222939	0.001618488093	
		Joule		Joule		Joule	

Figure 8: Table examples of Performance and Power statistics

The energy consumption was calculated by the formula below:

$$Energy\ Consumption\ in\ joules = Runtime\ Dynamic\ Power \times Execution\ time\ sim\_seconds$$

3 Results

3.1 Performance vs Frequency

The following figures are the performance analysis for matmul and crc32 respectively, using various frequencies and memory systems.

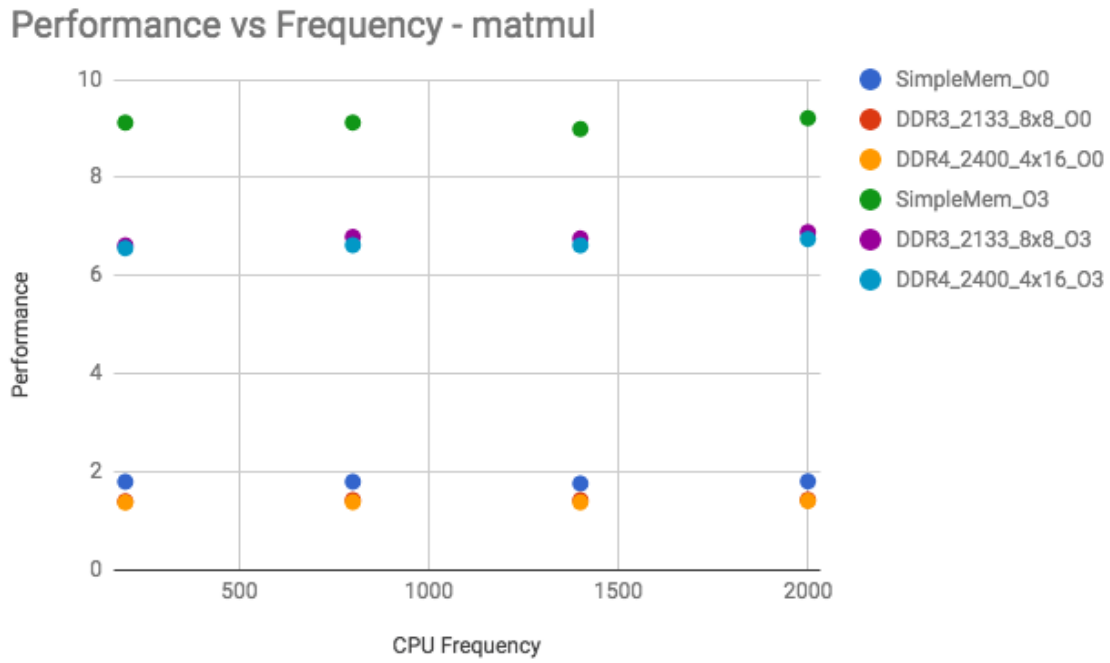

Figure 9: Gem5 simulation results using various memory systems and frequencies – matmul benchmark

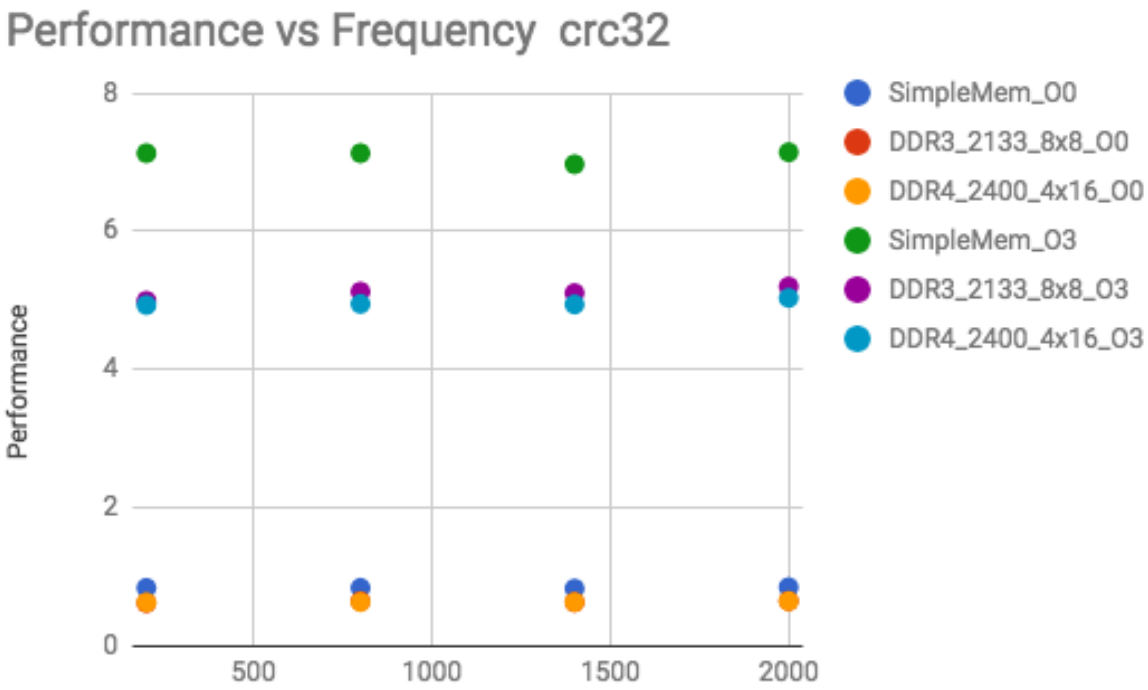

Figure 10: Gem5 simulation results using various memory systems and frequencies - crc32 benchmark

In both cases, SimpleMem_O3 outperforms all other configurations. The configuration with DDR4_2400_4x16_O0 runs slowest. In general, for the same configuration the simulation runs faster using matmul benchmark than crc32 benchmark.

3.2 Energy vs Frequency

The following figures are the energy consumption analysis for matmul and crc32 respectively, using various frequencies and memory systems.

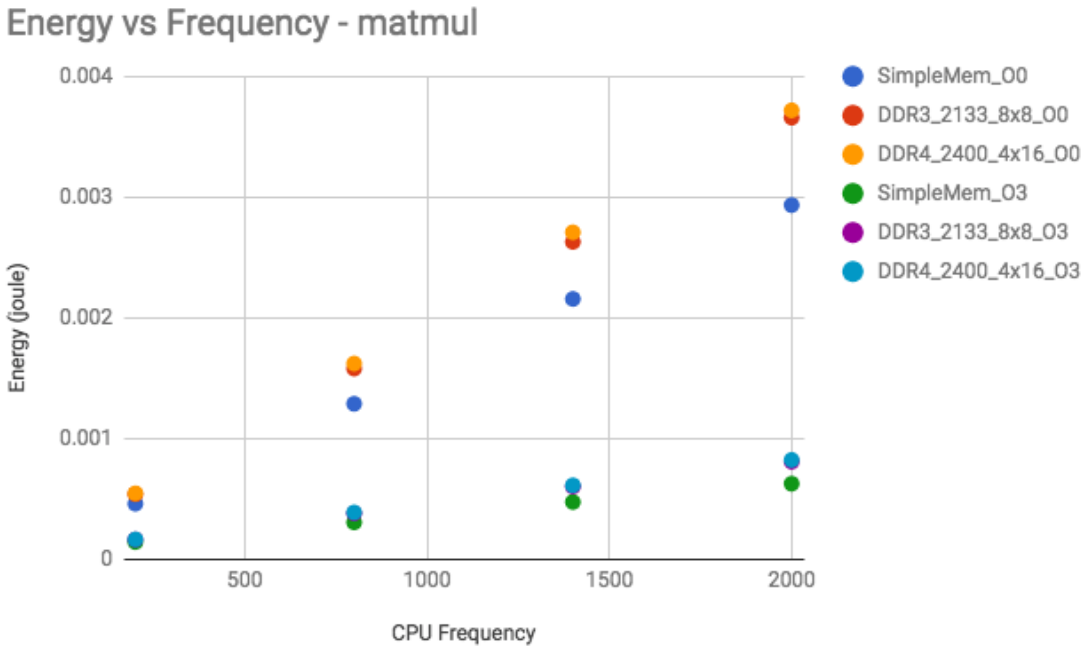

Figure 11: McPAT energy consumption result - matmul benchmark

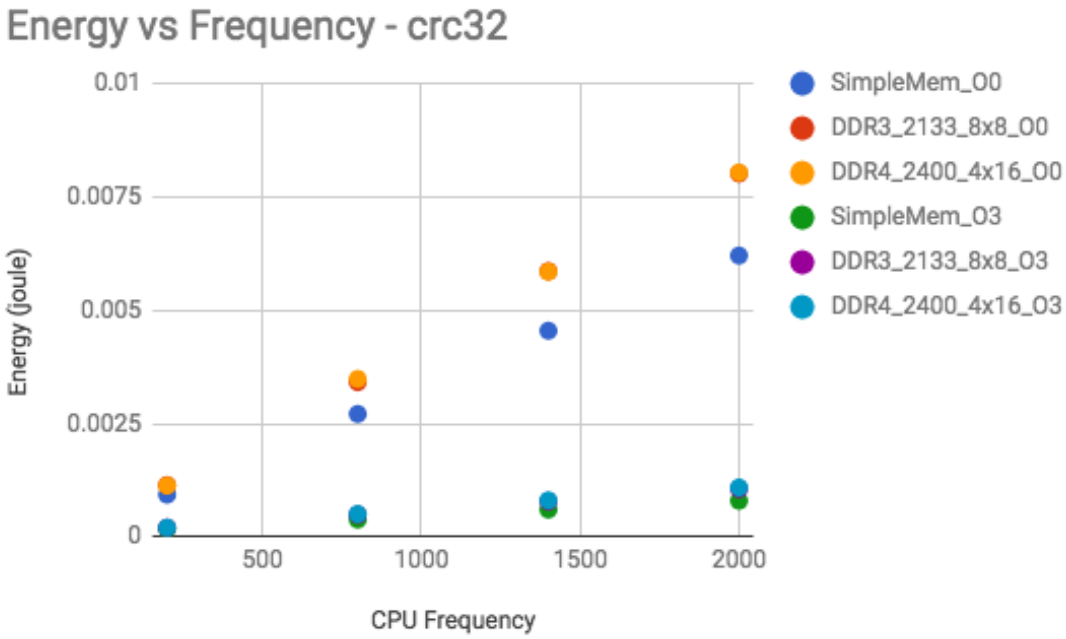

Figure 12: McPAT energy consumption result - crc32

In both cases, SimpleMem_O0 consumes the least amount of energy, and DDR4_2400_4x16_O0 consumes the most amount of energy. In general, both energy consumption increases linearly with increasing frequencies.

3.3 Energy vs Performance

The following figures shows the energy vs performance trend.

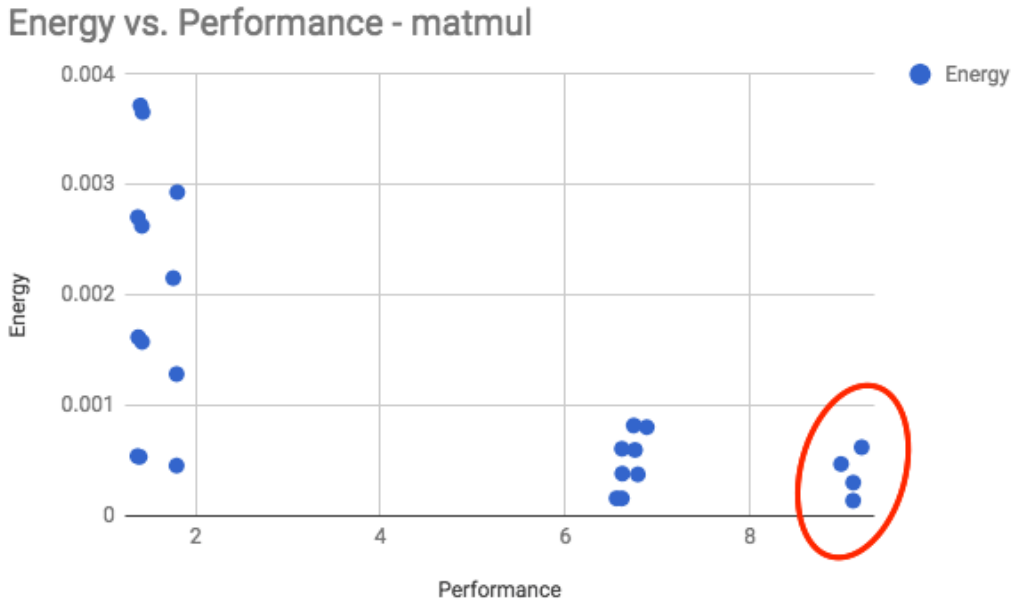

Figure 13: Energy vs Performance plot for matmul benchmark

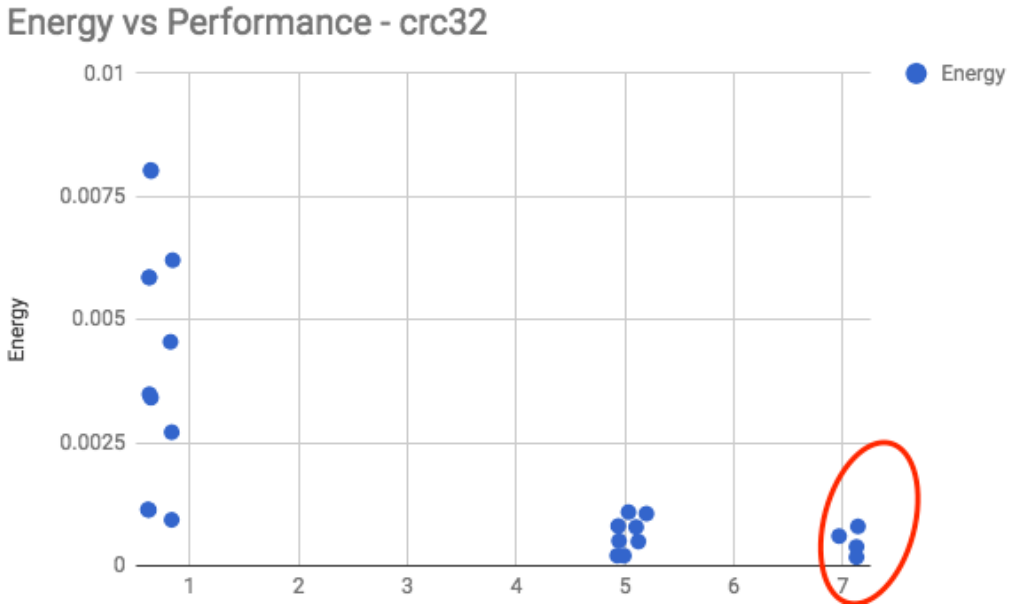

Figure 14: Energy vs Performance plot for crc2 benchmark

Unfortunately the Google spreadsheet does not have feature to add data label to the points. The user has to hover the mouse cursor on the point to identify it.

Base on Pareto optimization of multi-objective optimization, the points circled in red are the most suitable configurations for low energy consumption while still delivering high performance. These points are SimpleMemory configurations.